Manipulation des layouts

version 1

Interface Homme-Machine : Unity

Voici les objectifs de ce sujet :

— continuer à manipuler l'IDE `Unity` ;
— continuer la création d'un *widget* complexe ;
— exploiter les mécaniques vues précédemment ;
— utiliser les `Layouts`.

1 Description générale du TP

La fois précédente, nous avons réalisé nos premiers widgets complexes en exploitant l'agrégation de plusieurs widgets de base. Nous avons en particulier manipulé le système d'ancre que propose `Unity` pour placer les objets de façon relative.

Ici, nous allons voir une autre méthode un peu plus coûteuse mais offrant une plus grande puissance en terme de placement et qui reprend les points que vous avez étudié dans les années précédentes en IHM : les mises en page (`layout`). La documentation est présente ici : `https://docs.unity3d.com/Packages/com.unity.ugui@1.0/manual/UIAutoLayout.html`

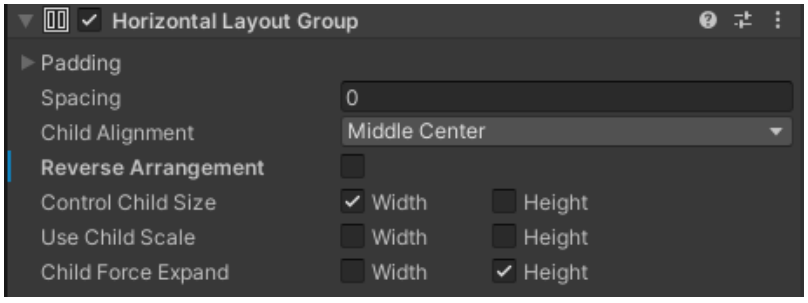
Composants basiques

— Tester les propriétés du composant `Content Size Fitter` sur un `GameObject` de type `Text` ou `Image`.
— Faites de même pour le composant `Aspect Ratio Filter`. Pour bien distinguer les effets de chaque composant, il est recommandé d'en associer un seul à la fois au `GameObject`.

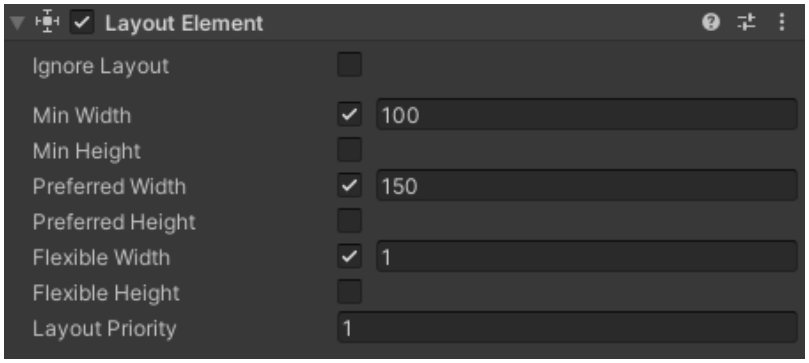
2 Présentation des outils de mise en pages en Unity UI

Il est conseillé de lire en détail la documentation citée plus haut pour bien comprendre les aspects et tous les détails. Voici le résumé des points clés :

— Il y a deux types de composants dédiés à la mise en page.
— Les *conteneurs* ou `Layout Group` contrôlent le comportement des widgets fils (enchaînement horizontal, enchaînement vertical ou sous forme de grille...)



— Les composants *éléments* ou `Layout Element` qui indiquent leur présence de mise en page.



Ainsi, chaque conteneur peut avoir des paramétrages différents qui vont faire évoluer les éléments selon les contraintes. Vous ferez attention à certains paramètres qui peuvent forcer le redimensionnement des `widgets` éléments sans les consulter. Vous ferez aussi attention aux éléments qui doivent activer la flexibilité des dimensions voulues pour agrandir dans cette direction (une valeur de 1 peut être suffisante pour indiquer un degré de liberté).

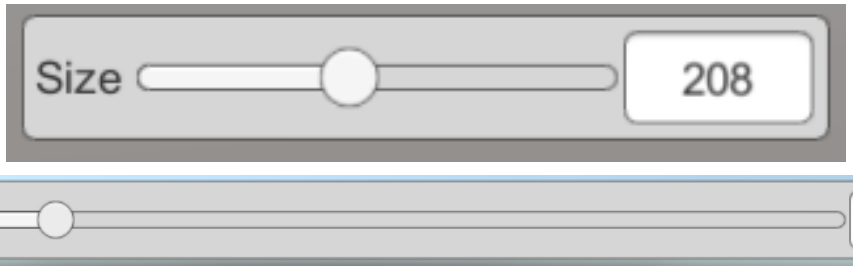
Tests

1. Dans un premier temps, tester les conteneurs `Horizontal Layout Group`, `Vertical Layout Group` et `Grid Layout Group` **sans** ajouter de composant `Layout Element` aux enfants des groupes.
2. Associer ensuite des `Layout Element` aux enfants des groupes et tester les conteneurs.

3 Widget : ComplexSlider - le retour en joli

Refaites le widget `ComplexSlider` (soit sous un autre nom, soit après avoir fait une sauvegarde de votre ancien projet/-widget). Pour obtenir un affichage joli, peu importe la largeur que vous donnerez à votre widget, pourvu que le `Slider` prenne la plus grande place.

Veillez à ce que le widget contenant la valeur numérique ne soit pas modifiable directement par l'utilisateur.



4 Widget : Spinner

Réalisez le widget du `Spinner` qui consiste à contrôler les évolutions d'un nombre via deux boutons regroupés en bout de ligne.

La valeur de l'incrément (positif ou négatif) sera laissé au choix de l'utilisateur.

Le widget doit être fonctionnel, mais vous pouvez bien sûr changer/adapter selon vos souhaits le coté esthétique du widget. De nouveau, vérifiez que le widget contenant la valeur numérique ne soit pas modifiable directement par l'utilisateur.

5 Aspects avancés sur le système d'événement souris

Nous présentons ici une mécanique pour interagir avec des événements particuliers. Pour cela, consultez le lien suivant qui donne les événements supportés `https://docs.unity3d.com/Packages/com.unity.ugui@1.0/manual/SupportedEvents.html`. Nous vous proposons un petit exercice sous forme de tutoriel :

— Dans un nouveau projet ou une nouvelle `Scene` [1], ajoutez un `Panel` occupant l'entièreté du `Canvas`.

— Modifiez la couleur du `Panel` pour qu'il soit entièrement transparent.

— Ajoutez un nouveau script.

— Au début du fichier script, ajoutez la ligne

1. Dans ce dernier cas, ajouter dans chaque scène un bouton pour permuter entre les scènes.

```
using UnityEngine.EventSystems;
```

— Faire hériter le script avec les événements voulus (cf. lien plus haut). Dans notre exemple, nous allons nous concentrer sur les cliques souris et donc nous prendrons l'interface : `IPointerClickHandler`.

— Surchargez les fonctions associées aux interfaces. Ici :

```
public void OnPointerClick(PointerEventData data) { }
```

Normalement, les événements gérés par cette mécanique réagissent par défaut et vous pouvez le vérifier avec des messages de `Log`.

A présent, nous souhaitons réaliser les traitements suivants.

1. Lorsque nous cliquons sur une zone de l'écran, nous voulons créer à la volée un `widget` de notre choix à l'écran en tant que fils de notre `Panel` initial (n'oubliez pas qu'un `widget` UI doit avoir comme parent un `Canvas`). On suppose que dans le script, on connaît à l'avance le `widget` qui sera cloné. Pour réaliser cela, consultez la page `https://docs.unity3d.com/ScriptReference/Object.Instantiate.html`.

2. Enfin, vous êtes prêt à créer un script `ResizeWidget` qui consiste à agrandir en largeur/hauteur un `widget` quelconque en faisant un drag sur ce `widget`.